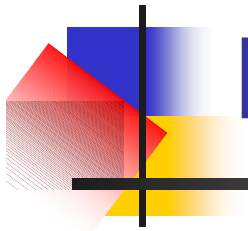
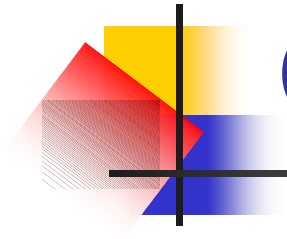


Unified Parallel C on BlueGene/L

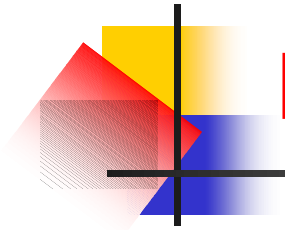
Rajkishore Barik
Calin Cascaval
Siddhartha Chatterjee (sc@us.ibm.com)
Maria Eleftheriou
Manish P. Kurhekar
Pradeep Verma

IBM Research (India, Yorktown)



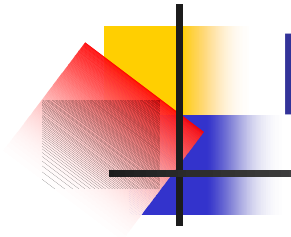
Outline of Talk

- Unified Parallel C (UPC): How, What, Why?
- IBM Research's involvement with UPC implementation



Unified Parallel C
History

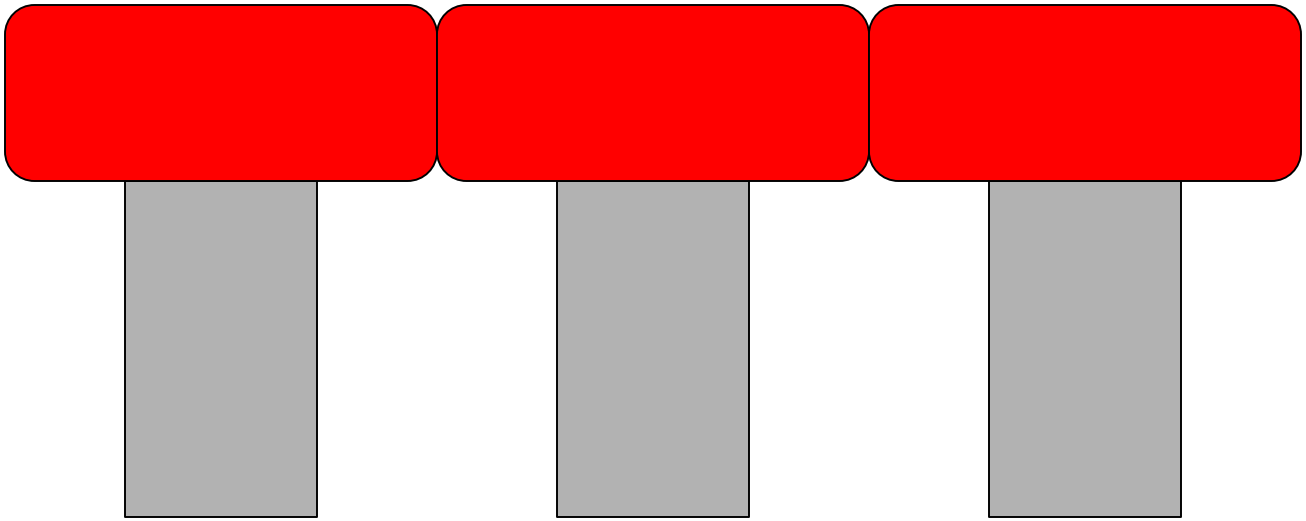
- Start with C (ISO/IEC 9899:1999 standard)
- Add parallelism
  - Unify experiences from Split-C, AC, PCP, etc.
- Initial tech report (IDA, LLNL, UCB) May 1999
- First implementations on Cray T3D/E (*gcc* based)
- Integrate developer and user experience
  - Government: ARSC, IDA CSC, LBNL, LLNL, NSA, US DOD
  - Academia: GWU, MTU, UCB
  - Vendors: Compaq, CSC, Cray, Etnus, HP, IBM, Intrepid Technologies, SGI, Sun Microsystems



Parallel Programming Models

- Programmer's view of data and control
  - Not the same as parallel architectures
  - Enables architecture-neutral programming if model can be mapped efficiently on real machines
- Examples
  - Message passing model (MPI)
  - Data-parallel model (HPF)
  - Shared memory model (OpenMP)
  - Distributed shared memory model (UPC)
  - Hybrid models (OpenMP + MPI)

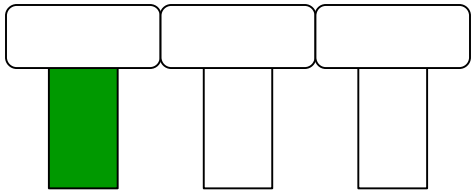
Overview of UPC
Distributed Shared Memory Model



Each thread can access data resident in:

- Local part of address space
- Shared part of address space with affinity to that thread
- Shared parts of address space with affinity to other threads

Programming Example
Vector Addition in C

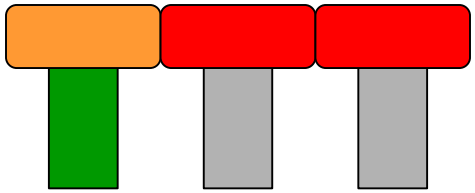


```
#define N 1000

        double v1[N], v2[N], v3[N];

void main() {
    for   (int i=0; i < N; i++;  )
        v3[i] = v1[i] + v2[i];
}
```

Programming Example
Vector Addition in UPC



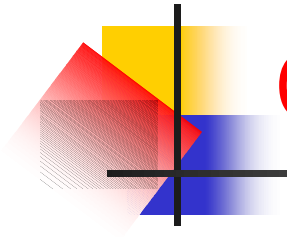
```
#include <upc_relaxed.h>
#define N 1000

shared double v1[N], v2[N], v3[N];

void main() {
    forall(int i=0; i < N; i++; i)
        v3[i] = v1[i] + v2[i];
}
```

UPC Features
Data

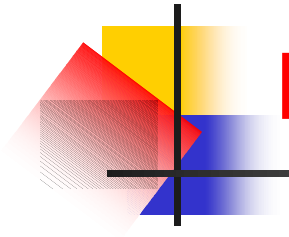
- Shared variables
  - `shared T x;`
  - Scalars, records, arrays (of ...), pointers (to ...)
  - Accessible to all UPC threads
- Affinity
  - `shared[10] float x[100];`
  - Can be dynamic
- Consistency
  - `[strict | relaxed] shared int x;`
  - Governs cross-thread observability of shared variable update
- Self knowledge
  - `THREADS`, `MYTHREAD`



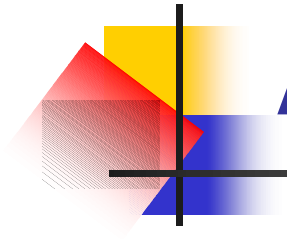
UPC Features
Control

- Parallel iteration
  - `upc_forall(j = 0; j < n; j++; j)`
  - Adds scheduling clause to normal C for-statement
    - Various possibilities
    - Implications for nested foralls
- Global synchronization
  - Barrier, wait/notify, fence

UPC Features
Libraries



- Current
  - General utilities
  - Dynamic memory allocation
  - Locks
  - Shared string handling
- Under development
  - Collective communication operations
  - I/O

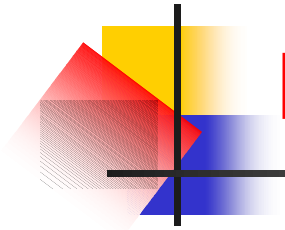


Assessment of UPC

- Pros
  - Extension of familiar C syntax
  - Locality exploitation: blocking, affinity
  - Implementations on several platforms
  - Incremental code parallelization
- Cons
  - Semantic extensions sometimes unintuitive
  - Affinity mechanism is limited
  - Implementations are limited
  - Small user community

http://www.gwu.edu/~upc

Our UPC Activity
Design Strategy



- Working with XL code base
  - Intent: Code changes merged back
- Changes required to normal XL compiler
  - Front end modification to handle new syntax
  - New pass to convert UPC semantics to C semantics + RTS calls
  - Linker modifications to handle memory segments
- RTS design
  - Trying to avoid scalability limitations of other implementations